Bach in a Box: The Evolution of Four Part Baroque Harmony Using the Genetic Algorithm

Ryan A. Mc Intyre
2151 Oakley Ave.
Menlo Park, CA 94025
rmcintyr@us.oracle.com

The space of all possible musical compositions given the constraints of the standard Western tonal and rhythmic system is uncountably vast. In this paper, we look at a small subsection of this space: Baroque harmony. The rules governing Baroque harmony have been carefully laid out by musical scholars, and the size of this search space becomes tractable if we are given a melody and a key signature to work with. Music is easily represented in numerical form, and is thus an obvious candidate for computer manipulation. In this paper, we investigate a system using the genetic algorithm that is capable of generating well constructed Baroque-style harmonies given a user-defined melody.

Introduction

Many different approaches and modes of thought have surfaced in the field of computer music. Some enthusiasts believe computers to be as capable as humans in generating interesting music. Others distrust computers and question whether they have any place in the emotional and creative world of music. The spectrum ranges from pieces of music composed in their entirety by a computer to the use of the computer as little more than electronic staff paper. Connectionist approaches have been used to generate melodies within a certain musical style after the network has been trained with examples from a particular type of music (Todd, 1989). Other approaches have used rule-based and grammar-based systems in the analysis and generation of music (Roads, 1979). We aim to use the computer as an assistant composer. The composer provides the melody, which is the important centerpiece of any composition, and the system will generate harmonies to accompany the melody. The genetic algorithm has been used as a composer's assistant before; work by Horner and Goldberg has investigated the use of genetic algorithms for thematic bridging (Horner and Goldberg, 1991). Thematic bridging involves transforming some melody into another melody through the use of intermediate differing but related melodies. The "Bach In a Box" system aims to be somewhere in the middle: it relies on a pre-defined melody as a structure around which it generates the harmonies. Given the design of the system, it would have been equally easy to have the melody be randomly generated, but this would have been a less useful tool to a composer who seeks to harmonize a melody he or she has written. Constraining the system to a pre-defined melody also allows for useful comparisons between evolutionary runs, and allows the composer to investigate many possible harmonization options for a single melody.

Background

Before the implementation of this problem can be discussed, it is important to understand the basics of the musical system and of baroque harmony. The main division of sound in music is the octave. When one sound has a pitch that is twice the frequency of another it is called an octave. For example, sounds playing at 440hz and at 880hz are an octave apart from one another. Western music divides the octave into 12 tones called half-steps. Seven letters are used to name the tones, with the sharp sign and the flat sign naming the other intermediate tones. So, in one octave, we have the following tones: a, a sharp/b flat (these two tones are just different names for the same pitch), b, c, c sharp/d flat, d, d sharp/e flat, e, f, f sharp/g flat, g, g sharp/a flat, and a. The a's at the beginning and end of the list are an octave apart from one another. Any notes that have the same name are an octave apart. Given the twelve tones in an octave, Western music has favored a particular pattern of seven notes per octave. This particular pattern is called the major scale and is the basis for most music that you hear. The pattern happens to be whole-step, whole-step, half-step, whole-step, whole-step, whole-step, half-step. If we look at our list of the 12 tones and start our major scale on a c, we see that we get the following seven tones: c, d, e, f, g, a, b. This conveniently avoids any tones that have sharps or flats. (These seven tones also correspond to the white keys on a piano. Ask any beginning musician which scale they prefer. It will always be C-major. Any other scale contains sharps or flats which are harder to read in musical notation.) The major scale defines the key signature. If we are using the C-major scale, we can say we are playing in the key of C.

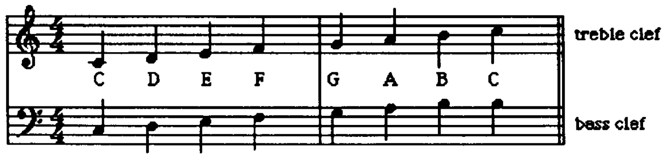


figure 1. C-major scale represented in treble and bass clef

Simple Baroque harmony works within the confines of the major scale. For simplicity's sake, we will only use the C-major scale for discussion. Once we have a scale, we can build chords, which are several notes played together at one time. Chords are built from triads. A triad comes from three notes: the root, the third, and the fifth. For example, a C-major triad is spelled C-E-G. A D-minor triad is spelled D-F-A. The reason the C triad is called major and the D triad is called minor has to do with the number of half-steps between the root and the third, but this subtlety need not be discussed here. There is a second type of triad called the diminished triad, and a second kind of chord called a dominant chord, which has four tones: the root, the third, the fifth, and the seventh. In any given key, only one chord can be dominant, and that chord is the fifth of the scale. So, in C-major, there can only be a G-dominant chord, and it is spelled G-B-D-F. We represent each chord in the a key with a roman numeral. We use capital letters for major chords, lower case for minor chords, a degree sign for the diminished chord and a 7 to notate the dominant chord. To summarize, in the key of C, we have the following: I (C-major), ii (D-minor), iii (E-minor), IV (F-major), V (G-major), V7 (G-dominant), vi (A-minor), and vii^{o} (B-diminished).

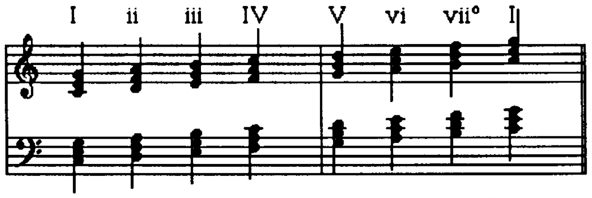


figure 2. triads and their labels in the key of C.

Now that we have described the musical system that we are working in, we can describe Baroque harmony. The study of Bach's work and the work of other composers from the Baroque era (1650-1750) including Vivaldi and Handel has yielded a set of general rules and regularities that are seen in examples of the style. Four part Baroque harmony involves the creation of four sequences of notes that work together to provide chord changes that are both interesting a pleasant to listen to. These four voices are soprano, alto, tenor, and bass. A four part Baroque harmony is most often played on a pipe-organ or sung by a choir, and provides a simple approach to harmonization that is used as the basis for teaching students the fundamentals of music theory. Most importantly, the four notes that are played simultaneously must form a legal chord within the key. Since four notes are being played, and there are three notes per triad, one of the notes must be doubled. In Baroque harmony, it is most desirable to double the root of the chord. Second choice doubling, in which the fifth is doubled, is less desirable. In most cases, the third is not doubled. The notes need not be played in any particular order. Thus, any set of tones including the notes C, E, and G is a C-major chord. Which note the soprano, alto, tenor, or bass is playing is unimportant, As long as all the notes in the chord are accounted for. There are yet more constraints placed on each chord. In general, there should not be too much distance between adjacent voices. Intervals larger than an octave between adjacent voices sound strange. And, it is desirable that the total distance between the soprano and bass voices be 3 octaves or less.

Once the chords meet these conditions, we can look at the movement between chords. The analysis of inter-chord movement is called voice leading. Most desirable in voice leading is contrary motion between soprano and bass and between alto and tenor. If the notes moves up in pitch in one of the members of the soprano-bass, alto-tenor pairs, the other should move down in pitch. If contrary motion is not present, oblique motion is desirable, in which one of the pair stays static, and the other moves. Least desirable is parallel motion. Beyond voice leading, the smoothness of each individual line is important. Each of the lines should be examined individually. A line is smooth if the distance in pitch between each adjacent note in a line is small. This is more pleasant to listen to than a choppy line that has wide-pitch intervals between consecutive notes. The last factor that needs to be examined in a Baroque harmony is resolution. Certain chords, such as the V, the V7, and the vii^{o}, lead naturally to the I chord, and sometimes the vi chord. That is, upon hearing one of these chords, our ears "expect" to hear a I chord following them. Once we have a proper chord resolution, we need to look closer to see if we can perform what is called a "tritone" resolution. Notice that the V7 and vii^{o} in C-major each contain the notes B and F. These two notes form what is called a "tritone". (Which also has to do with the number of half-steps between them, but is beyond the scope of this discussion.) These two notes have a tension in them, and if the B moves to the adjacent C, and the F moves to the adjacent E, we have a proper tritone resolution. Notice that the C and E are present in both the I chord and the vi chord in C-major. This is why these chord must follow the V7 and vii^{o} chords if tritone resolution is to occur. The rules of **chord spelling, doubling, voice leading, smoothness,** and **resolution** make up most of basic Baroque harmony. As more of these devices are applied, we get a harmony that is intricate, well-crafted, and in the Baroque style.

The Problem

In order to make this an interesting tool, it was decided that the system would take a melody as its input and generate the lines of the other three voices to create the four part harmony. As was decided in the above discussion, the melody and harmony were constrained to the key of C-major. This ignores interesting features of baroque harmony such as secondary dominants, pivot chords, and key changes, but as an initial system designed to test the feasibility of the genetic algorithm as applied to Baroque harmony, it is quite sufficient, and allows for a thorough exploration of the rules listed above. We use a four-octave range, starting with the C below the bass clef up to the C two lines above the treble clef. This gives us 29 possible note values within the key of C. An integer ranging from

0 to 28 is used to represent each pitch. To represent a four-note chord, we need a group of four integers. Thus the integers 7,4,2,0 represent the notes C, E, G, and C, which is a C-major triad. If we have a string of 8 integers, we have two consecutive chords represented. Therefore, the following string represents a transition from a C-major to a D-minor chord: 7,4,2,0,8,5,3,1. The first position (7) and the fifth position (8) represent the two notes in the soprano line. The tenor line is represented in the third position (2) and seventh position (3). The alto and bass lines are similarly present. In musical terms, the soprano moves from a C to a D, while the tenor moves from an E to an F. To further clarify the representation scheme, an example is given from the text-file output of the system, with the same harmony in musical staff notation presented to its right.

```
Best of Generation:
Soprno: g3/25  b4/27  a4/26  f3/24  e3/23
Alto:   c3/21  d3/22  e3/23  d3/22  c3/21
Tenor:  e2/16  f2/17  e2/16  g2/18  g2/18
Bass:   c2/14  g1/11  c2/14  b2/13  c2/14
        I      V7     vi     V7     I
fitness=0.971429
```

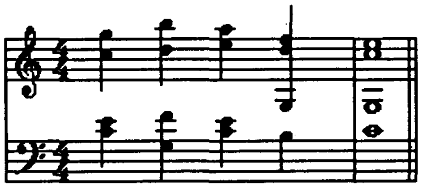


figure 3. outputfile representation of individual and musical notation representation

Listed in sequence, this string is: 25,21,16,14,27,22,17,11,26,23,16,14,24,22,18,13,23,21,18,14. A string of integers is all that is needed to represent a four part harmony. To simplify analysis of the individual musical pieces, the user defined melody was included in all of the individuals in the population. As the individuals were randomly generated, the melody was copied into the proper positions in the string, depending on what voice it was specified to be in. In the above example, the user defined melody appears in the alto line. To represent a four part baroque harmony given a melody of length n, we need an individual with chromosome length of n*4. Above, we have a 5 note melody. The result is a chromosome of length 20. Given this information, we can calculate the size of our search space. We have an alphabet of size 29, and a chromosome of length (4*melody). For a given melody of length n, we have a search space of size $29^{(n*4)}$.

The fitness measure analyzes each individual on the basis of chords, ranges, motion, harmonic interest, the begin and end chords, smoothness and resolution. Each of these sub measures awards points to the harmony in a range from 0 to some multiple of 4 multiplied by the length of the user defined melody. The maximum possible raw fitness an individual could have would be 68 * (length of melody). After the raw fitness is calculated, it is divided by the maximum possible value, which insures that the fitness will fall on a real numbered value between 0 and 1. Each of the components of the fitness function scans through the chords or harmony and melody lines and award single points at a time based on how well the structure matches the criteria in the component of the fitness function. There is some maximum number of points a given individual can earn, and this is determined by the length of the melody. In the following examples, "length" always refers to the length of the user defined melody and not to the length of the chromosome, which is four times longer than the melody. The component functions are as follows:

goodChords - returns a score from 0 to 16*length. Points are awarded for the proper spelling of a chord and for proper doubling of the chord.

goodRanges - returns a score from 0 to 16*length. Points are awarded for proper spacing between voices as well as for the preservation of the melody line. For example, if the melody is in the alto voice, two chord tones must have pitches below the melody, and one must sound above the melody.

goodMotion - returns a score from 0 to 8*length. Points are awarded at a higher value for contrary motion, and at a lower value for oblique motion. Parallel motion is not awarded.

goodStartStop - returns a score from 0 to 4*length. Points are awarded for starting and ending the harmony on a I chord or a vi chord. These chords define the key signature and serve as good start and stop points.

harmonicInterest - returns a score from 0 to 8*length. Points are awarded for a varying chordal structure. Award begins at an automatic 8*length. Points are subtracted when two adjacent chords appear that have the same analysis, such as two I chords or two iii chords appearing side by side. This keeps the harmony from becoming boring.

smoothness - returns a score from 0 to 8*length. Points are awarded when adjacent notes in each of the four lines do not shift more than a fourth in interval. This prevents the song from sounding disjointed.

goodResolutions - returns a score from 0 to 8*length. Points are awarded for V7, V, and vii^{o} chords that are followed by I chords. Points are also awarded for proper tritone resolution down to the I or vi chords from the V7 and vii^{o} chords.

We can see that these component functions return a maximum possible sum of 68*length. The master fitness function sums these raw values and divides them by 68*length, yielding a standardized fitness between 0 and 1. An important attribute to note about all of theses functions is that they return graded values. None of them only return 0 or the maximum value. This way, harmonies and chords that are almost correct or that have a small amount of contrary motion are awarded in a graded fashion. This insures that promising genetic material will have a high fitness value.

However, since the most important component of a harmony is proper chord formation, fitness is awarded in a stepwise, three-tiered fashion. An individual must receive at least 85% of its possible maximum fitness from the goodChords and goodRanges functions before it is awarded fitness for any of the other component functions.

Then it must receive 85% of its possible points from the goodMotion, goodStartStop, and harmonicInterest function before it can receive fitness from the smoothness and goodResolutions functions. These divisions are not arbitrary, but rather reflect the importance of a good chordal foundation before voice leading is worth considering, and the importance of good voice leading and chord formation as necessary prerequisites before such factors as smoothness and resolution can be considered. Initial attempts at evolution not using this stepwise method failed to evolve reasonable individuals, since an individual without a single harmonically correct chord can still have perfect voice leading and smoothness of line, which can lead the evolution in the wrong direction. It is important to first develop a base of individuals who have proper chord spellings, and continue from there.

The standard operators of reproduction, crossover, and mutation are used. Mutation is modified to randomly change a note value in the string to any of the other 29 possible values. Since we have confined our musical problem to four octaves in the key of C-major, we can effectively search any conceivable four part harmonization of a given melody. Thus, the entire search space is accessible.

The initial population is randomly generated, and each slot in the string taking on a value from 0 to 28. Before the evolution begins, however, the user-defined melody is grafted into the appropriate points in the string, in order to assure that the melody is preserved between individuals during reproduction and crossover. Mutation is prevented from altering the melody. The option of grafting the user defined melody into each chromosome was chosen rather than using the chromosome to represent only the harmony voices because the calculation of fitness was much simplified and easily generalized when the melody was included in each individual.

Results/Discussion

The system was developed in C, using a modified version of Goldberg's simple genetic algorithm, which appeared in Pascal. Runs were made on Sun SparcStations and DEC 5000's. Population size ranged from 2000-4000, and were run from 400-600 generations. Runtime for a harmonization of a 5-9 note melody was 1-3 hours. However, these times reflect running the processes on heavily used public workstations with multiple users logged in. A dedicated machine would no doubt provide shorter runtimes.

A simple example run on a five note melody in the soprano voice follows, in order to illustrate the workings of the system. The arrow to the right of the soprano voice indicates that the user defined melody appears in the soprano voice. Population size was 2000, and it was set to run for 200 generations. The following individual demonstrates and individual whose fitness has not escaped from the first tier of the stepwise fitness function.

```
Gen= 1 max=0.3706 min=0.1765 avg=0.2563 sum=512.6058 nMut=322 nCross=784
Best of Generation:
Soprno:          c3/21  d3/22  e3/23  d3/22  c3/21   <-
Alto:            a4/26  d3/22  b2/13  g3/25  a2/12
Tenor:           c2/14  a3/19  g1/11  g2/18  f1/10
Bass:            f1/10  f2/17  f1/10  b1/ 6  c1/ 7
                 IV     ii     NC     V      IV
fitness=0.370588
```

Notice the NC symbol below the third chord. This means "No Chord." The third group of four notes does not form a legal chord. Each individual line lacks smoothness, and there are points where the alto voice is given a note that is higher than the soprano. Also note that song begins and ends on the IV, which is undesirable.

In the intermediate stages of this runs, we get individuals who have entered the second and third tiers of the fitness function.

```
Gen=43 max=0.8824 min=0.2882 avg=0.6065 sum=1213.0283 nMut=280 nCross=791
Best of Generation:
Soprno:          c3/21  d3/22  e3/23  d3/22  c3/21   <-
Alto:            g2/18  b3/20  g2/18  f2/17  g2/18
Tenor:           e2/16  f2/17  b1/ 6  a2/12  g1/11
Bass:            g1/11  g0/ 4  e0/ 2  d1/ 8  e0/ 2
                 I      V7     iii    ii     I
fitness=0.882353
```

Notice that all the chords are properly formed and that the voices all fall into their proper ranges. The individual also begins and ends on the I chord, for which it gains fitness. The few factors that prevent this individual from having higher fitness are the lack of smoothness in the tenor and bass lines and the non-resolution of the V7. Baroque harmony requires a I chord after a V7, but here we have a iii.

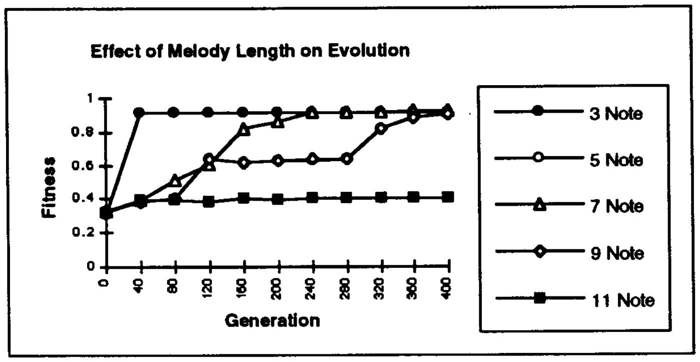
In the final stages of the evolution we see some sophisticated individuals. This particular run converged to a high fitness early, so we have a fitness above .90 before generation 100.

```
Gen=77 max=0.9118 min=0.3000 avg=0.7340 sum=1467.9940 nMut=318 nCross=816
Best of Generation:
Soprno:          c3/21  d3/22  e3/23  d3/22  c3/21   <-
Alto:            g2/18  b3/20  g2/18  b3/20  g2/18
Tenor:           e2/16  g1/11  b2/13  f2/17  e2/16
Bass:            g1/11  f1/10  e1/ 9  g1/11  g1/11
                 I      V7     iii    V7     I
fitness=0.911765
```

Here we have contrary motion, smoothness of lines, and a V7 to I resolution. All we are lacking is a tritone resolution in the final two chords and a proper V7 resolution following the second chord. A final comment is that the tenor voice could be a bit smoother that it currently is. Otherwise, this is a very good example of a Baroque harmony. Unfortunately, this run did not yield a significantly better individual after after this point.

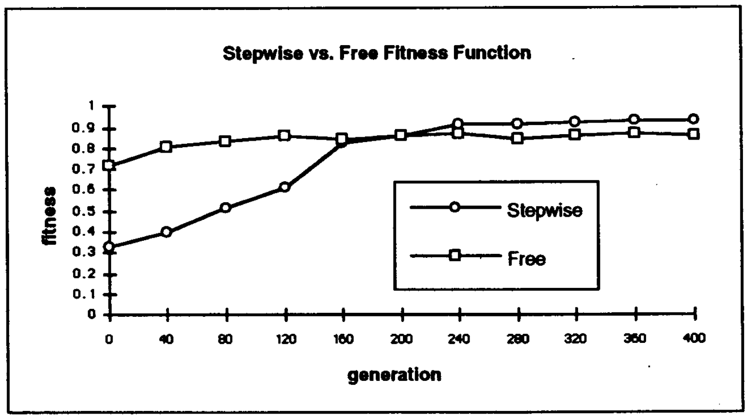
Experimental runs were made to compare the number of generations until an acceptable individual was produced given melodies of different length. An acceptable individual is defined as an individual with fitness above 0.93. Other runs were made to determine the affect on evolution of moving a constant melody among each of the four voices. Experimental evolutions made to determine the effect of the placement of melody in different voices were inconclusive and seemed to suggest that the placement of the melody voice does not make a large impact on the success or rate of the overall evolution. Finally, experiments were also run to evaluate whether a non-stepwise fitness function can produce useful results.

The following data were obtained from the evolutions of differing length melodies. Runs lasted for 400 generations, and used a population size of 2000.



Note that although the difference between the three note and eleven note runs were appreciably different, the data from the inner runs with five, seven, and nine note melodies were noisy. More evolutions need to be performed in order to make this data statistically significant, but one could reasonably expect that, given a constant yet sufficient population size, the lines would separate, with the shorter melodies peaking first, since the size of the search space is much smaller.

The most convincing experiments proved that the stepwise fitness function produces faster, more efficient evolutions. In all cases, the stepwise fitness function achieved higher fitness best-of-run individuals than the free-non constrained fitness function did. And, in all cases, the free fitness function produced higher fitness individuals in the initial stages. However, the fitness stayed even, while the stepwise fitness soon overtook the free fitness function. The following graph gives an example of one such run.



The above graph also clearly illustrates the major stumbling block that was encountered during the development of Bach in a Box. The initial version of the fitness function did not have the stepwise feature, and evolutions were not able to develop individuals with proper chord structure and harmonic placement. No matter how high the fitness got, best-of-generation individuals appeared with a note in the alto voice higher than a note in the soprano voice, or with an illegal combination of notes that did not form a chord. Many runs simply degenerated, with fitness staying at an even level. The stepwise fitness function solved this problem by insuring that only individuals with proper chordal structure could receive fitness for the "finer points" of Baroque harmony. It should also be noted that musically, this graph does not reflect a fair comparison. The stepwise fitness function outperforms the free function more significantly than is shown on this graph. The scale for the stepwise fitness is more rigorous, given that a high fitness individual must contain at least harmonically correct chords. An individual produced by the free function might have the same fitness level as one produced by the stepwise function, but the free fitness individual will likely still contain malformed chords and other mechanical problems that will make it far less pleasant to listen to.

Conclusion

The genetic algorithm proved a useful tool for generating four part Baroque harmony. For any given melody, many different high fitness individuals were produced in a series of evolution. Nearly every evolution produced a believable harmony, and the differences in the harmonizations of the same melody gave the genetic algorithm an edge over conventional rule-based systems that might only produce one or two different harmonizations for a given melody. The non-linear constraints placed on the formation of harmonies in the Baroque style proved to be a well suited problem for the genetic algorithm.

There is certainly room for more work in this area. The fitness function could be refined to include more subtle points of Baroque harmony that were not addressed in this paper, or were simply unknown to the author. Future work could be done to investigate whether certain melodies are harder to harmonize than others. The Bach in a Box system could be used to rate the difficulty a certain melody presents to a potential harmonizer. Since it searches the space of all possible harmonies, melodies that are difficult to harmonize in the Baroque would presumably have fewer maxima in the search space and would thus take more time to discover.

Another obvious improvement would be to introduce sharps and flats into the key or to open up the search space to the full twelve tone per octave system. This would allow for key changes and other interesting harmonic devices. Finally, users familiar with other forms of music might create fitness functions that could create harmonies for Jazz or, more frighteningly, Barber Shop Quartet harmonies.

Acknowledgements

This paper was written as a project for John Koza's *Genetic Algorithms* class at Stanford University in the Spring of 1993. I am grateful for his valuable assistance and advice in the course of this project, and thankful for the opportunity to have taken the two most challenging and interesting classes of my studies at Stanford from him.

References

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley Publishing Company, Inc. 1989.

Horner, Andrew and Goldberg, David E. "Genetic Algorithms and Computer-Assisted Music Composition". *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Diego, CA: Morgan Kaufman Publishers, Inc. 1991.

Koza, John R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press. 1992.

Roads, Curtis. "Grammars as Representations for Music". *Computer Music Journal,* Vol 3, No. 1, 1979. Cambridge, MA: Massachusetts Institute of Technology. 1979.

Todd, Peter M. A "Connectionist Approach To Algorithmic Composition". *Computer Music Journal,* Vol. 13, No. 4, Winter 1989. Cambridge, MA: Massachusetts Institute of Technology. 1989.